Decision Trees

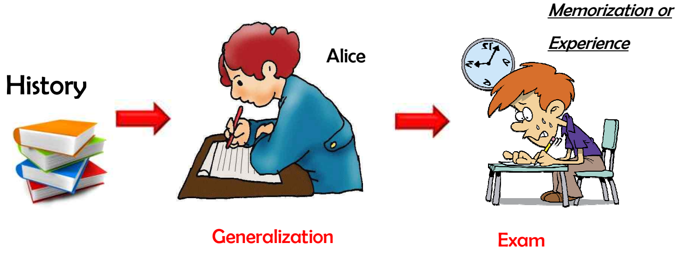
Machine Learning

Youngjoong Ko
Dept. of Computer Engineering,
Dong-A University

Contents

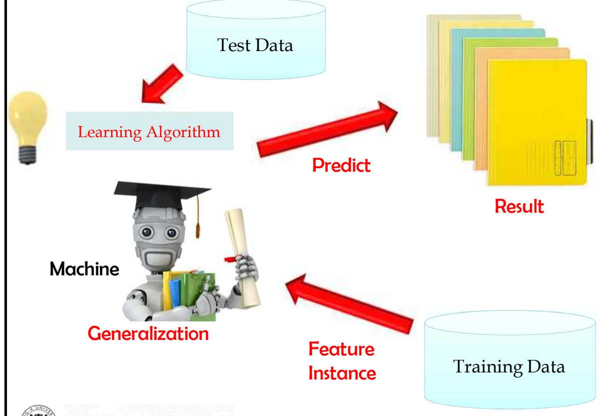
Machine Learning

❖ What does it mean to learn in human?



Machine Learning

❖ What does it mean to learn?

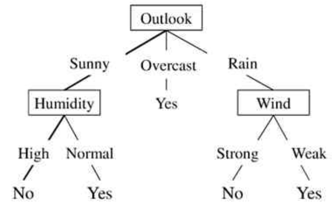


Canonical Learning Problems

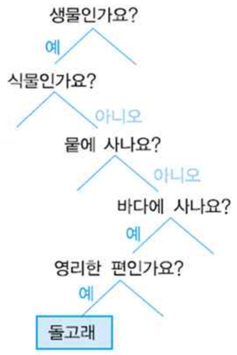
- ***Regression***: trying to predict a real value
  - predict the value of a stock tomorrow given its past performance
- ***Binary Classification***: trying to predict a simple yes/no response
  - predict whether Alice will enjoy a course or not
- ***Multiclass Classification***: trying to put an example into one of a number of classes
  - predict whether a news story is about entertainment, sports, politics, religion
- ***Ranking***: trying to put a set of objects in order of relevance
  - predicting what order to put web pages in, in response to a user query

Principle

- Examples of Decision Tree



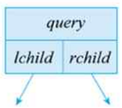
Tree



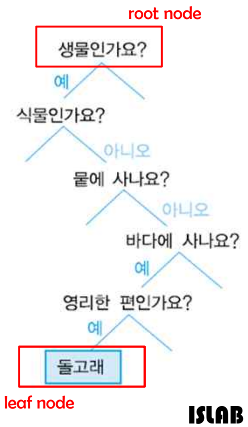
Binary Tree

Principle

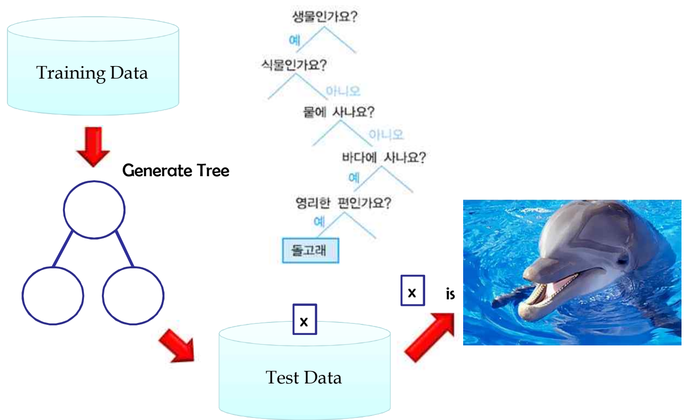
- If training data were given, then we would make the Decision Tree by learning from Training data
- Constraints
  - How get "question" from each node?
  - How well would I have done?
  - How leaf node decide class ?




```
struct node {
    struct question query;
    struct node *lchild;
    struct node *rchild;
};
```

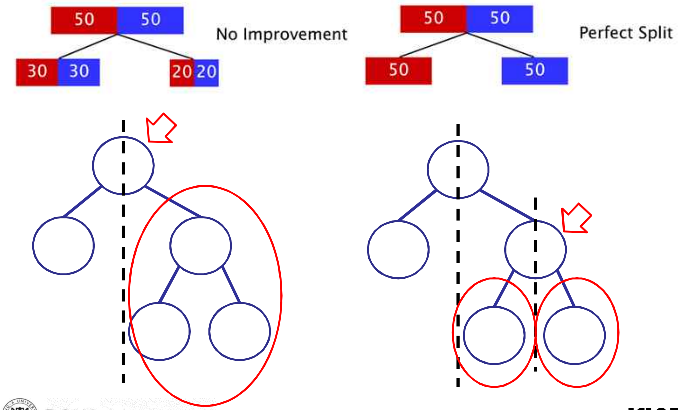


Progress



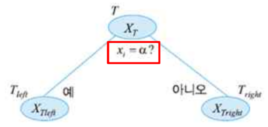
Impurity

❖ What's mean impurity?



Impurity

❖ **Question of node**



T is current node

X is set of features; d

a is value in feature x_i ; n

then

candidates of question; dn

Example : 혈액형이 b?

Entropy impurity : $im(T) = -\sum_{i=1}^{M} P(\omega_i \mid T) \log_2 P(\omega_i \mid T)$

Gini impurity : $im(T) = 1 - \sum_{i=1}^{M} P(\omega_i \mid T)^2 = \sum_{i \neq j} P(\omega_i \mid T) P(\omega_j \mid T)$

Misclassification impurity : $im(T) = 1 - \max_i P(\omega_i \mid T)$

$$P(\omega_i \mid T) = \frac{X_T\text{에서 } \omega_i\text{에 속한 샘플의 수}}{|X_T|}$$

Example

❖ Table of sample set X_T

샘플	특징벡터			부류
	x_1(직업)	x_2(선호 품목)	x_3(몸무게)	
1	3	1	50.6	w_2
2	2	3	72.8	w_1
3	3	5	88.7	w_3
4	2	2	102.2	w_2
5	5	5	92.3	w_2
6	3	4	65.3	w_2
7	2	3	67.8	w_1
8	7	1	47.8	w_3
9	2	3	45.6	w_1

Example

❖ Compute Impurity

$$X_T = \{(\mathbf{x}_1,\omega_2), (\mathbf{x}_2,\omega_1), (\mathbf{x}_3,\omega_3), (\mathbf{x}_4,\omega_2), (\mathbf{x}_5,\omega_2), (\mathbf{x}_6,\omega_2), (\mathbf{x}_7,\omega_1), (\mathbf{x}_8,\omega_3), (\mathbf{x}_9,\omega_1)\}$$

$$P(\omega_1 \mid T) = 3/9,\ P(\omega_2 \mid T) = 4/9,\ P(\omega_3 \mid T) = 2/9$$

Entropy impurity : $im(T) = -(\frac{3}{9}\log_2\frac{3}{9} + \frac{4}{9}\log_2\frac{4}{9} + \frac{2}{9}\log_2\frac{2}{9}) = 1.5305$

Gini impurity : $im(T) = 1 - (\frac{3^2}{9^2} + \frac{4^2}{9^2} + \frac{2^2}{9^2}) = 0.642$

Misclassification impurity : $im(T) = 1 - \frac{4}{9} = 0.556$

Example

❖ Generate candidate question

직업 (x_1): [1,7]의 정수 (1 = 디자이너, 2 = 스포츠맨, 3 = 교수, 4 = 의사, 5 = 공무원, 6 = NGO, 7 = 무직)

선호 품목 (x_2): [1,5]의 정수 (1 = 의류, 2 = 전자 제품, 3 = 스포츠 용품, 4 = 책, 5 = 음식)

몸무게 (x_3): 실수

➢ questions is :

x_1에 의한 후보 질문: x_1=1?, x_1=2?, x_1=3?, x_1=4?, x_1=5?, x_1=6?, x_1=7?

x_2에 의한 후보 질문: x_2=1?, x_2=2?, x_2=3?, x_2=4?, x_2=5?

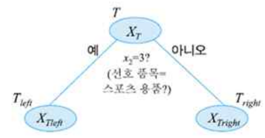
45.6, 47.8, 50.6, 65.3, 67.8, 72.8, 88.7, 92.3, 102.2

x_3에 의한 후보 질문: x_3<46.7?, x_3<49.2?, x_3<57.95?, x_3<66.55?, x_3<70.3?, x_3<80.75?, x_3<90.5?, x_3<97.25?

Example

❖ Impurity decrement

$$\Delta im(T) = im(T) - \frac{|X_{Tleft}|}{|X_T|} im(T_{left}) - \frac{|X_{Tright}|}{|X_T|} im(T_{right})$$



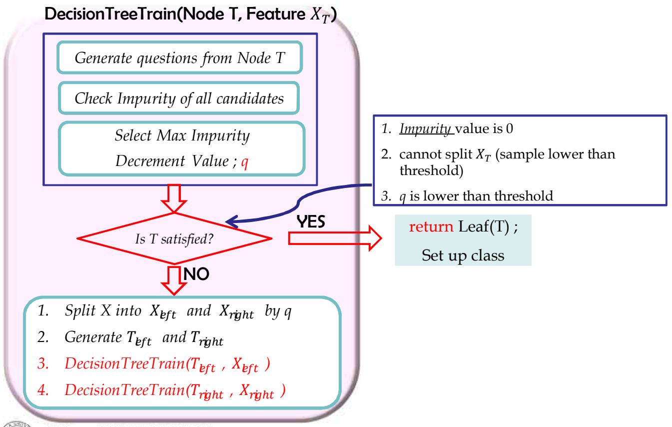
X_T={($\mathbf{x}_1,\omega_2$), ($\mathbf{x}_2,\omega_1$), ($\mathbf{x}_3,\omega_3$), ($\mathbf{x}_4,\omega_2$), ($\mathbf{x}_5,\omega_2$), ($\mathbf{x}_6,\omega_2$), ($\mathbf{x}_7,\omega_1$), ($\mathbf{x}_8,\omega_3$), ($\mathbf{x}_9,\omega_1$)}

X_{Tleft}={($\mathbf{x}_2,\omega_1$), ($\mathbf{x}_7,\omega_1$), ($\mathbf{x}_9,\omega_1$)}

X_{Tright}={($\mathbf{x}_1,\omega_2$), ($\mathbf{x}_3,\omega_3$), ($\mathbf{x}_4,\omega_2$), ($\mathbf{x}_5,\omega_2$), ($\mathbf{x}_6,\omega_2$), ($\mathbf{x}_8,\omega_3$)}

$$\Delta im(T) = 0.642 - \frac{3}{9} * 0.0 - \frac{6}{9} * 0.444 = 0.346$$

Decision Tree Train

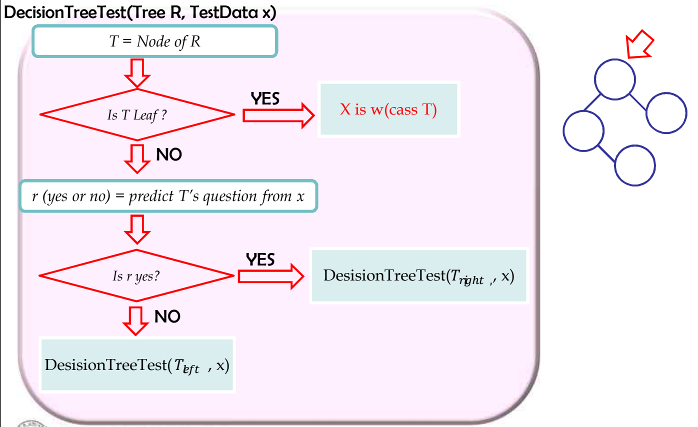


Decision Tree Train

Algorithm 1 DecisionTreeTrain(*data, remaining features*)

```
guess ← most frequent answer in data                  // default answer for this data
if the labels in data are unambiguous then
   return Leaf(guess)                                 // base case: no need to split further
else if remaining features is empty then
   return Leaf(guess)                                 // base case: cannot split further
else                                                  // we need to query more features
   for all f ∈ remaining features do
      NO ← the subset of data on which f=no
      YES ← the subset of data on which f=yes
      score[f] ← # of majority vote answers in NO
                 + # of majority vote answers in YES
                        // the accuracy we would get if we only queried on f
   end for
   f ← the feature with maximal score(f)
   NO ← the subset of data on which f=no
   YES ← the subset of data on which f=yes
   left ← DecisionTreeTrain(NO, remaining features \ {f})
   right ← DecisionTreeTrain(YES, remaining features \ {f})
   return Node(f, left, right)
end if
```

Decision Tree Test



Decision Tree Test

Algorithm 2 DecisionTreeTest(*tree*, *test point*)

```
if tree is of the form Leaf(guess) then
    return guess
else if tree is of the form Node(f, left, right) then
    if f = yes in test point then
        return DecisionTreeTest(left, test point)
    else
        return DecisionTreeTest(right, test point)
    end if
end if
```


Formalizing the Learning Problem

- there are several issues when formalizing the notion of learning
  - The performance of the learning algorithm should be measured on unseen "test" data
  - The way in which we measure performance should depend on the problem we are trying to solve
  - There should be a strong relationship between the data that our algorithm sees at training time and the data it sees at test time
- loss function
  - to tell us how "bad" a system's prediction is in comparison to the truth. In particular
  - if y is the truth and $\hat{y}$ is the system's prediction, then $\boxed{\ell(y, \hat{y})}$ is a measure of error.

Formalizing the Learning Problem

- For three of the canonical tasks discussed above, we might use the following loss functions

Regression: **squared loss** $\ell(y, \hat{y}) = (y - \hat{y})^2$
or **absolute loss** $\ell(y, \hat{y}) = |y - \hat{y}|$.

Binary Classification: **zero/one loss** $\ell(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$

Multiclass Classification: also zero/one loss.

  - Note that the loss function is something that you must decide on based on the goals of learning
- Now that we have defined our loss function, we need to consider where the data comes from

Formalizing the Learning Problem

- There is a probability distribution D over input/output pairs
  - This is often called the **data generating distribution**
  - If we write x for the input and y for the output, then D is a distribution over (x, y) pairs
- Formally, it's **expected loss** ϵ over D with respect to ℓ should be as small as possible
  - we don't know what D is!

$$\epsilon \triangleq \mathbb{E}_{(x,y)\sim D}\left[\ell(y, f(x))\right] = \sum_{(x,y)} D(x,y)\ell(y, f(x))$$

Formalizing the Learning Problem

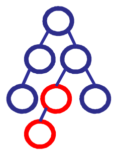
- Suppose that we denote our training data set by D
  - The training data consists of N-many input/output pairs, $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$
  - Given a learned function f, we can compute our **training error**

$$\hat{\epsilon} \triangleq \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n))$$

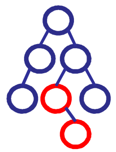
    - *our training error is simply our average error over the training data*
- Given a loss function ℓ and a sample D from some unknown distribution D, you must compute a function f that has low expected error ϵ over D with respect to ℓ

Inductive Bias

- *What we know before the data arrives*



Preference type A



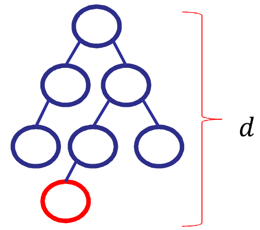
Preference type B

- Preference for one distinction over another is a bias that different human learners have
- **inductive bias**: in the absence of data that narrow down the relevant concept

Inductive Bias

- We will not allow the trees to grow beyond some pre-defined maximum depth, d
  - That is, once we have queried on d-many features, we cannot query on any more and must just make the best guess we can at that point
- The key question is: What is the inductive bias of shallow decision trees?
  - Roughly, their bias is that decisions can be made by only looking at a small number of features

Shallow decision tree

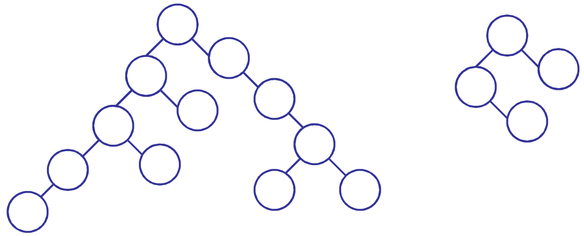


Not Everything is Learnable

- There are many reasons why a machine learning algorithm might fail on some learning task
- There could be noise in the training data
  - Noise can occur both at the **feature level** and at the **label level**
- Some example may not have a **single correct answer**
- In the **inductive bias case**, it is the particular learning algorithm that you are using that cannot cope with the data

Overfitting and Underfitting

- Overfitting is when you pay too much attention to idiosyncracies of the training data, and aren't able to generalize well
- Underfitting is when you had the opportunity to learn something but didn't.
  - This is also what the empty tree does



Overfitting Underfitting

Separation of Training and Test Data

- The easiest approach is to set aside some of your available data as "test data" and use this to evaluate the performance of your learning algorithm.
- If you have collected 1000 examples, You will select 800 of these as **training data** and set aside the final 200 as **test data**.
- Occasionally people use a 90/10 split instead, especially if they have a lot of data
- They **cardinal rule** of machine learning is:

Never *ever* touch your test data!

Models, Parameters and Hyperparameters

- The general approach to machine learning, which captures many existing learning algorithms, is the **modeling** approach
- For most models, there will be associated **parameters**
- **Hyperparameter** : we can adjust between underfitting and overfitting by the ***DecisionTreeTrain*** function so that it stop recursing
  - choosing hyperparameters: choose them so that they minimize training error
- The job of the **development data** is to allow us to tune hyperparameters

Development Data

- Some people call this "**validation data**" or "**held-out data**."
- Split your data into 70% training data, 10% development data and 20% test data
- For each possible setting of your hyperparameters
  - Train a model using that setting of hyperparameters on the training data
  - Compute this model's error rate on the development data
- From the above collection of models, choose the one that achieved the lowest error rate on development data
- Evaluate that model on the test data to estimate future test performance.